AMENDMENTS TO THE CLAIMS

The listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. A computer-implemented method of scheduling threads for a plurality of simultaneous multi-threading (SMT) ~~SMT~~ processors, said method comprising:

determining that a first thread in a first run queue is a poor performing thread, wherein the first run queue corresponds to a first SMT processor, wherein the determining further includes: [[;]]

executing a plurality of threads listed in the first run queue, including the first thread, on the first SMT processor, the executing further including:

retrieving a number of cycles value for each thread indicating the number of cycles that occurred while each thread was executing;

retrieving a number of instructions value for each thread indicating the number of instructions that were executed while each thread was executing;

dividing each number of cycles value by its corresponding number of instructions value, the dividing resulting in a cycles per instruction (CPI) value; and

recording the CPI value for each thread in the first queue; and

identifying the first thread as having a CPI value worse than a plurality of the other threads listed in the first run queue; and

in response to the determination:

writing a first identifier corresponding to the first thread to a second run queue, wherein the second run queue corresponds to a second SMT processor; and

removing the first identifier from the first run queue.

2. (original) The method as described in claim 1 further comprising:

determining that a second thread in the second run queue is another poor performing thread;

in response to the determination regarding the second thread:

writing a second identifier corresponding to the second thread to the first run queue; and

removing the second identifier from the second run queue.

3. (canceled)

4. (currently amended) The method as described in claim 1 [[3]] further comprising:

determining whether each thread was previously moved from one of a plurality of SMT processors' run queues to the first SMT processor's run queue.

5. (original) The method as described in claim 4 further comprising:

determining that the CPI of the first thread has degraded since being moved to the first SMT processor's run queue.

6. (currently amended) The method as described in claim 1 [[3]] further comprising:

recording the first thread's identifier, the first thread's CPI, and a timestamp to a previously swapped data structure.

7. (original) The method as described in claim 6 wherein the timestamp corresponds to a time selected from the group consisting of a time that the first thread's CPI was calculated, and a time that the first thread's identifier was moved from the first run queue to the second run queue.

8. (currently amended) The method as described in claim 1 [[3]] further comprising:

averaging the CPI value for each of the plurality of threads with one or more previous CPI values previously calculated for each of the threads, wherein the recorded CPI value includes the average CPI value for each thread.

9. (original) The method as described in claim 1 further comprising:

skipping one or more worse performing threads in comparison to the first thread in response to determining that each of the worse performing threads has previously been moved to the first SMT processor's run queue and each has improved in performance since being moved; and

identifying the first thread after skipping the worse performing threads.

10. (canceled)

11. (canceled)

12. (canceled)

13. (canceled)

14. (canceled)

15. (canceled)

16. (canceled)

17. (canceled)

18. (canceled)

19. (canceled)

20. (canceled)

21. (canceled)

22. (canceled)

23. (canceled)

24. (canceled)

25. (canceled)

26. (canceled)

27. (canceled)

28. (canceled)

29. (canceled)

30. (canceled)